TOSHIBA

Leading Innovation

meta-debian

Extending Yocto Project's Poky for building Debian-based embedded systems

Kazuhiro Hayashi, TOSHIBA Corporation
LinuxCon Japan 2015
Jun 5, 2015

Introduction

- **Linux is used all around the world**
- **We also use Linux in some of our products**
  - Including Social infrastructure[1]
- **There are many Linux distributions to choose from**
- **Things to consider**
  - The number of supported packages
  - Package versions
  - Supported hardware
  - Stability, number of bugs
  - The frequency of security updates and supported timespan
  - How to compile and customize packages

In our case

- **What we want to do**
  - Make custom embedded Linux environments
- **What we need**
  - Wide hardware support
  - Stability
    - Old but well tested packages are better new but unstable ones
  - Long-term support
    - Over 5 years support required
  - Fully customizable build system

Our solution

Yocto Project "poky"

- One of the most popular reference distributions for embedded Linux
- Fully customizable build system
- Supports numerous embedded boards including modern ones

Debian GNU/Linux

- Supports basic embedded CPUs: x86, ARM, PowerPC, MIPS (32bit/64bit)
- Releases a stable version after two years of testing
- Long-term support for 5 years

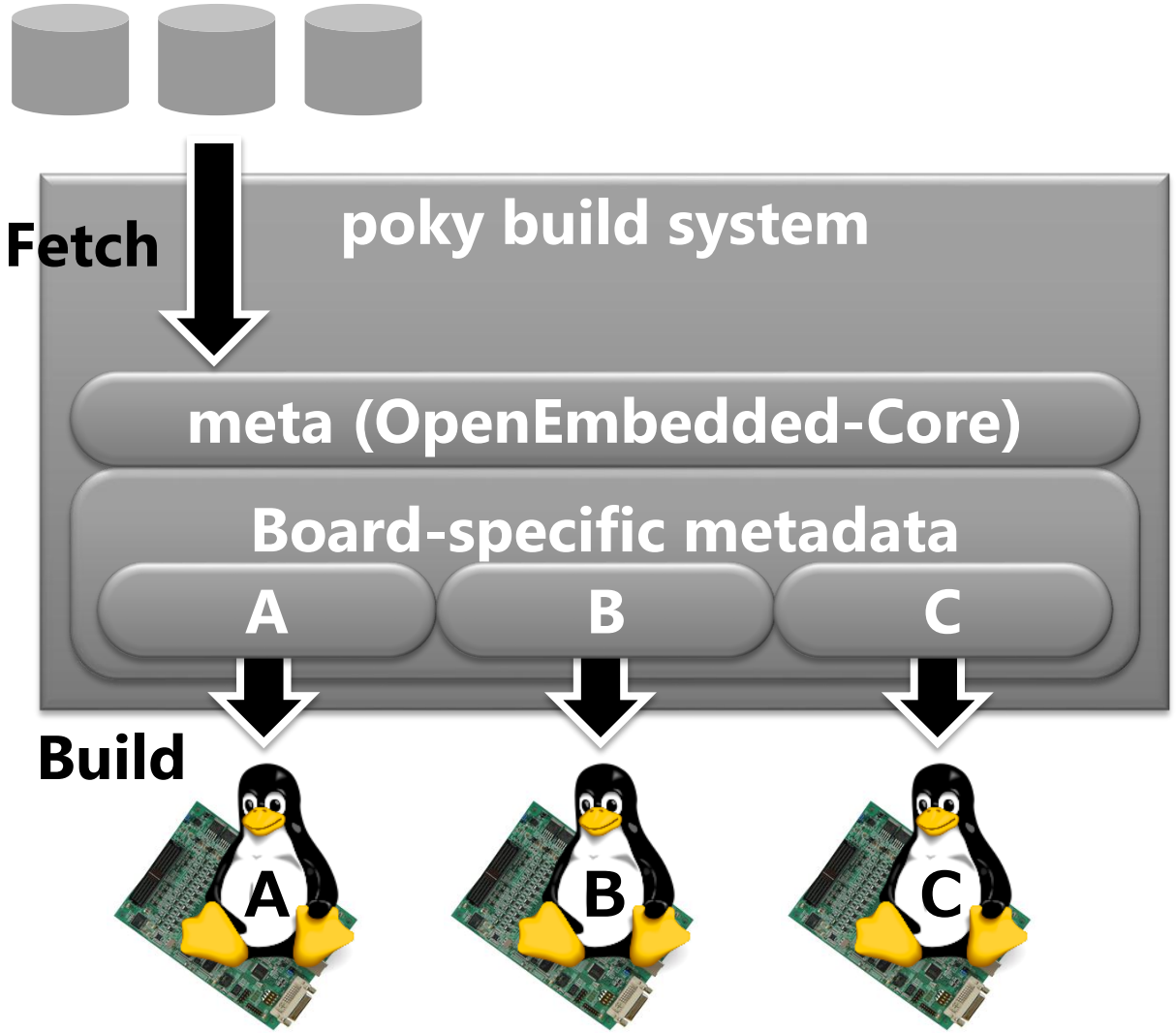
meta-debian

What is meta-debian?

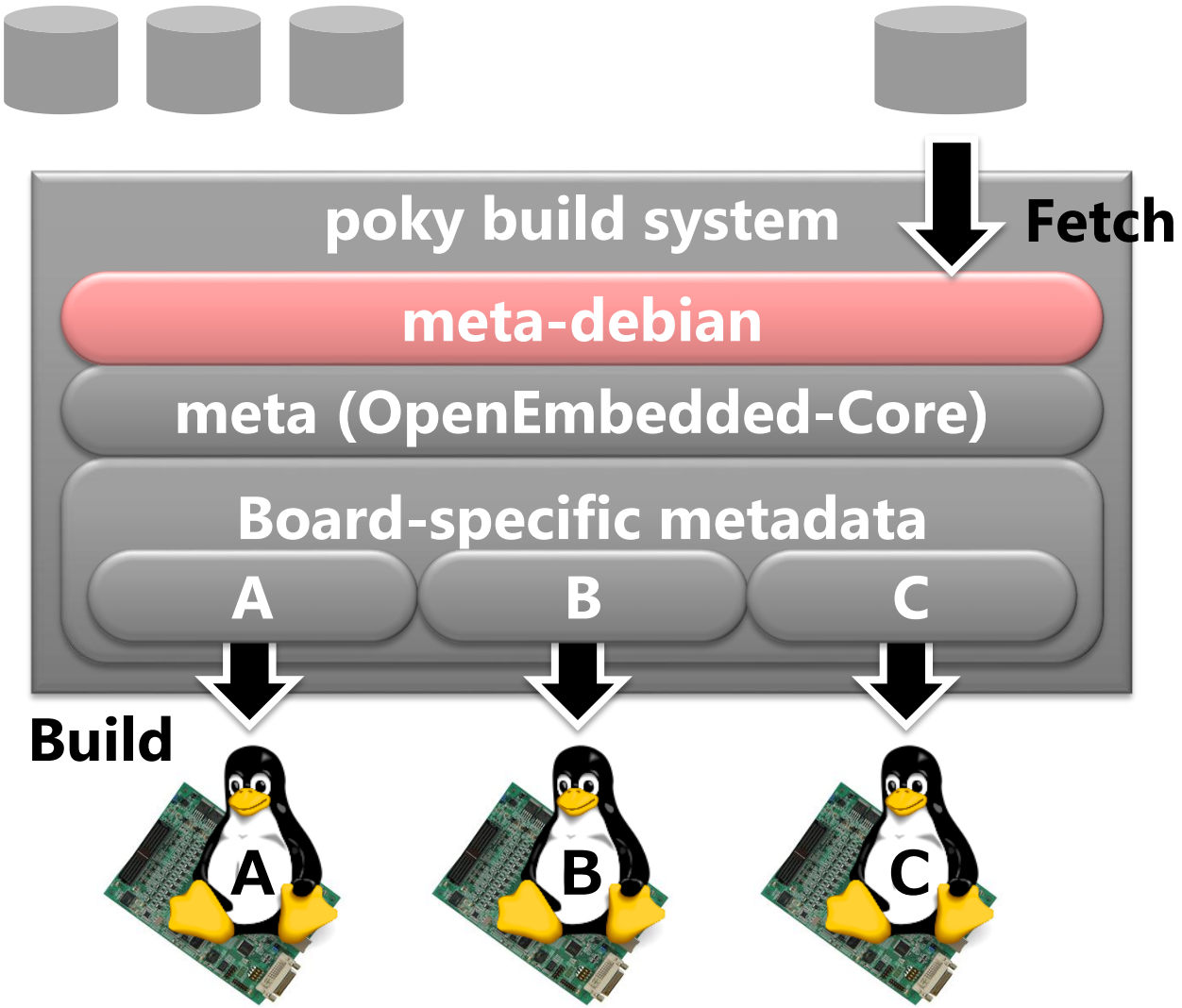
- **A set of recipes (Metadata) for the poky build system**
- **Main feature**
  - Allows cross-building Linux images using Debian source packages
- **Implemented as an independent "layer"**
  - Completely separated from OpenEmbedded-Core and other layers
  - Already registered in OpenEmbedded metadata index
    - http://layers.openembedded.org/layerindex/branch/master/layer/meta-debian/

Build system structure (poky)

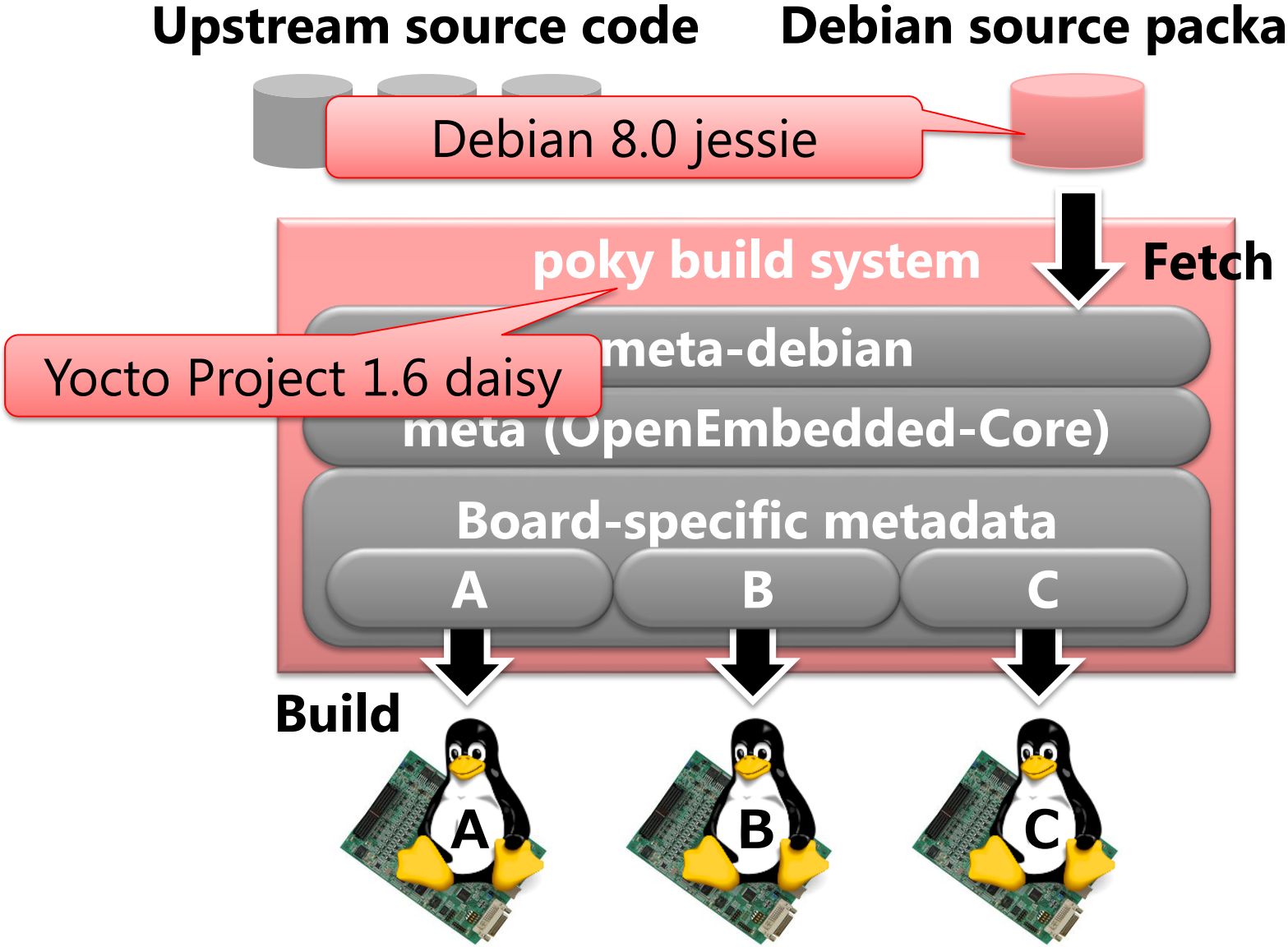
Upstream source code



Build system structure (poky + meta-debian)



Target versions of meta-debian



Purpose of meta-debian

- **Satisfy our customer needs**
  - Wide embedded CPU support
  - Stability
  - Long-term support
  - Fully customizable build system

With Debian stable release + LTS

With poky build system

- **Provide a common place for developers having the same needs**
- **Contribute to upstream (Debian and Yocto Project)**

Quick start

1. Download the build tools
2. Setup build directory
3. Build tiny Linux image
4. Run tiny Linux image on QEMU

- See also meta-debian/README

Download build tools

- **Download poky**

```
$ git clone git://git.yoctoproject.org/poky.git
$ cd poky
$ git checkout daisy
```

- **Download meta-debian into the poky directory**

```
$ cd poky
$ git clone https://github.com/meta-debian/meta-debian.git
$ cd meta-debian
$ git checkout daisy
```

- **← meta-debian specific step**

Setup build directory

- **Change the default configuration to meta-debian's**
  - Enable meta-debian layer
  - Enable "debian" distro (DISTRO = "debian")
  - The default target machine is "qemux86" (MACHINE = "qemux86")
  - TEMPLATECONF is used by oe-init-build-env script

```
$ export TEMPLATECONF=meta-debian/conf
```

- **Run startup script**
  - This setup a build directory and environment variables automatically
  - (builddir): name of build directory (optional)

```
$ source /path/to/poky/oe-init-build-env (builddir)
```

Build tiny Linux image

- **Run bitbake**

```
$ bitbake core-image-minimal
```

- **Built images (case of qemux86)**
  - Output directly
    - /path/to/builddir/tmp/deploy/images/qemux86
  - Kernel
    - bzImage-qemux86.bin
  - Root filesystem
    - core-image-minimal-qemux86.ext3
    - core-image-minimal-qemux86.tar.gz

Run tiny Linux image on QEMU

- **Run built images on QEMU environment**
  - qemux86

```
$ runqemu qemux86 nographic bootparams="init=/init root=/dev/sda"
```

  - qemux86-64

```
$ runqemu qemux86-64 nographic bootparams="init=/init root=/dev/sda"
```

  - qemuarm

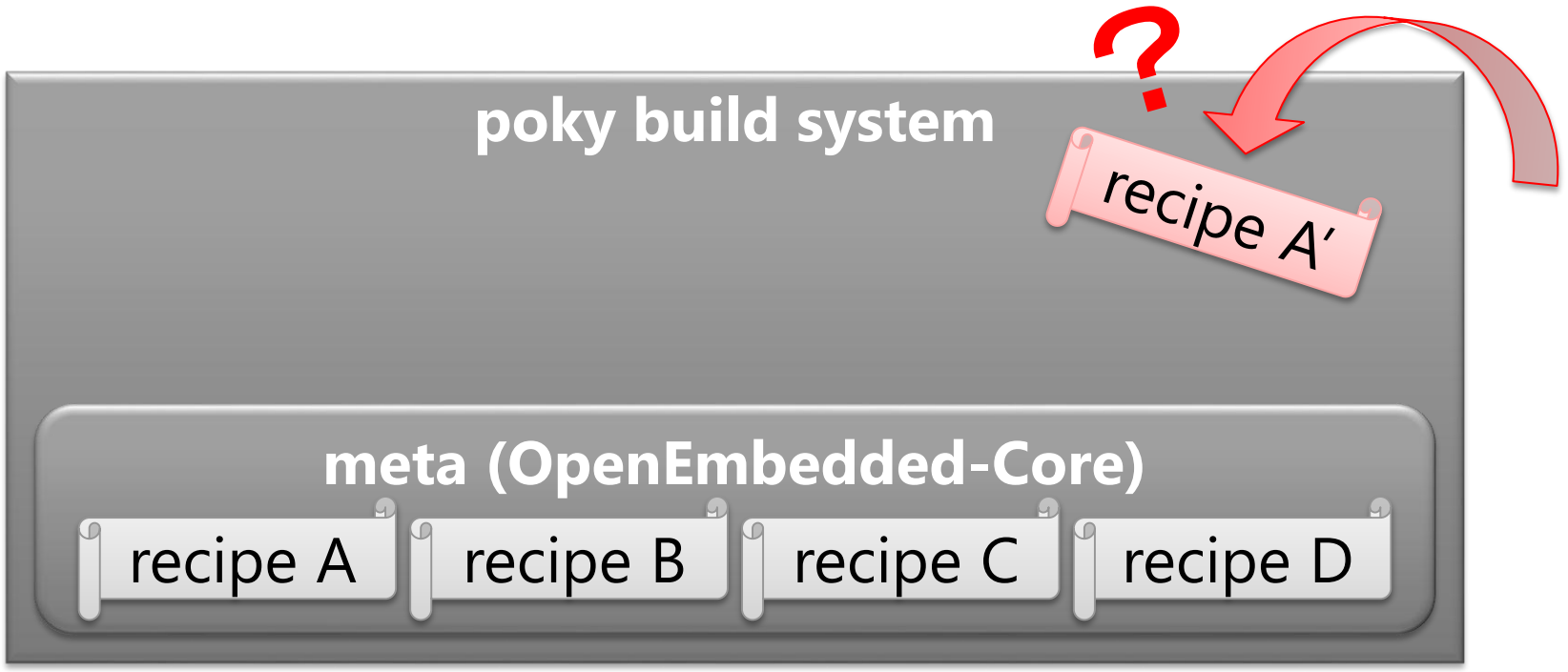
```
$ runqemu qemuarm nographic bootparams="init=/init console=ttyAMA0"
```

  - qemuppc

```
$ runqemu qemuppc nographic bootparams="init=/init"
```

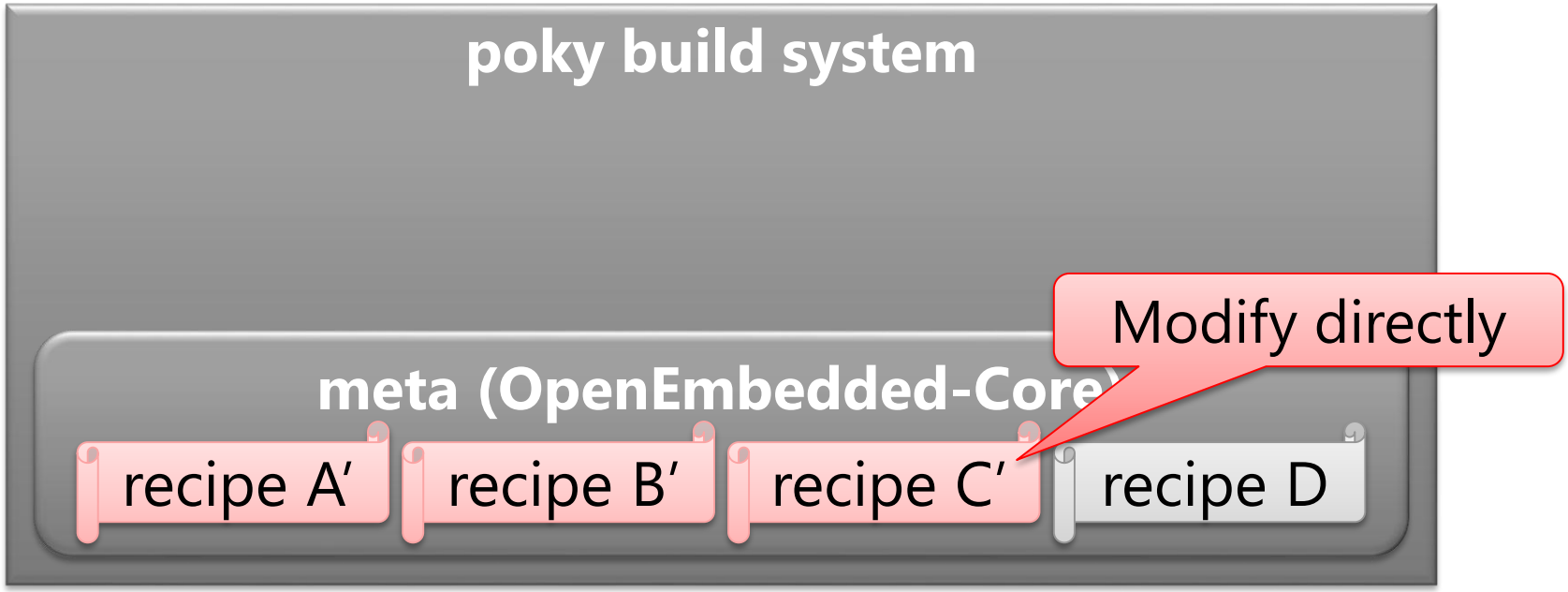
How should we create recipe files?

- **We need to create new recipes for Debian sources**
  - How?



Method 1: Modify OE-Core recipes

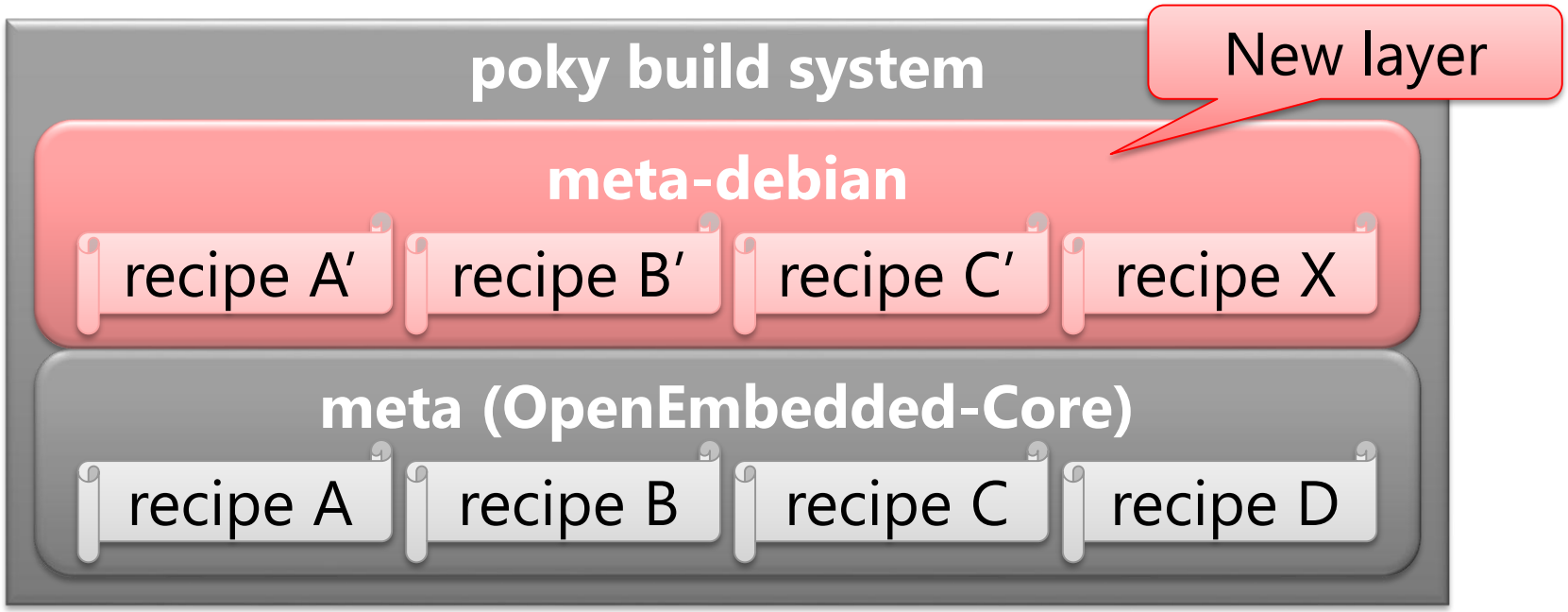
- **We already tried this way previously: "poky-debian"[2]**
- **Not the ideal solution ☹**
  - Original OE-Core recipes are no longer available
  - Just a fork
    - It becomes hard to catch up with the newest poky versions
    - Difficult to convince other people to join our effort



Method 2: Add recipes into a new layer

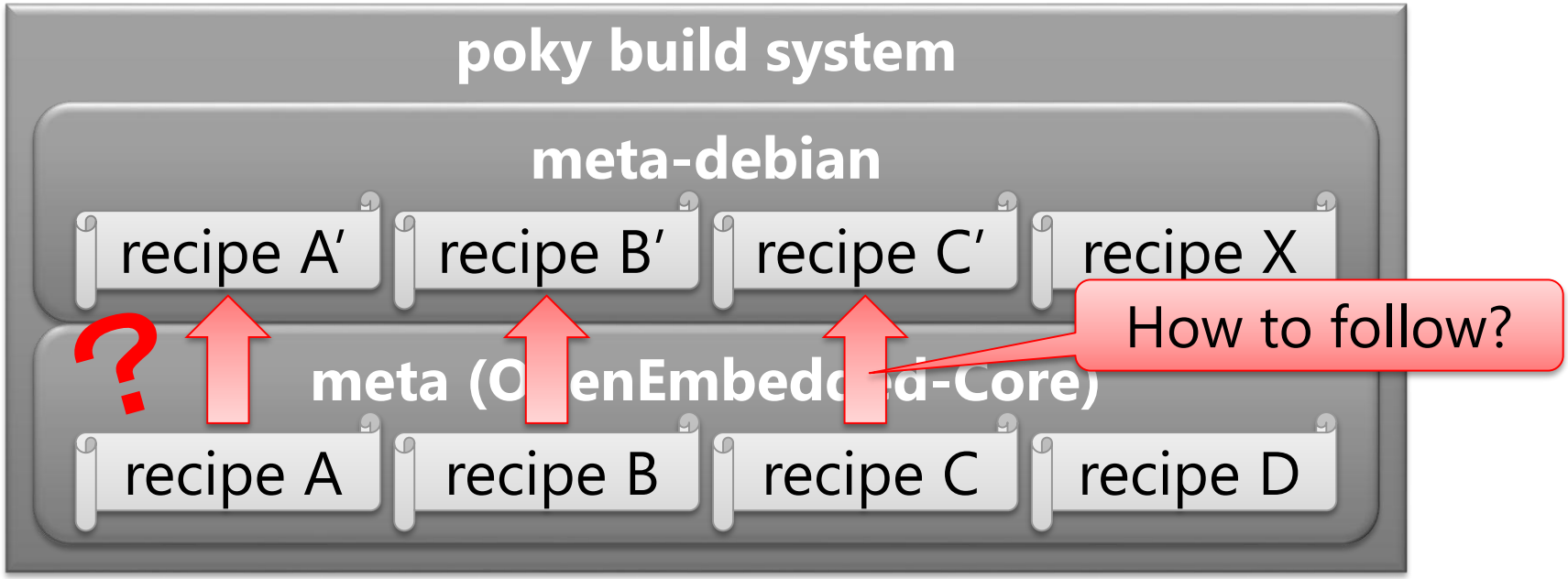
our solution

- **The best way to add new recipes for specific purposes**
  - Original OE-Core recipes are available
  - Can be developed independently of OE-Core
  - Enable / disable the layer easily like a module



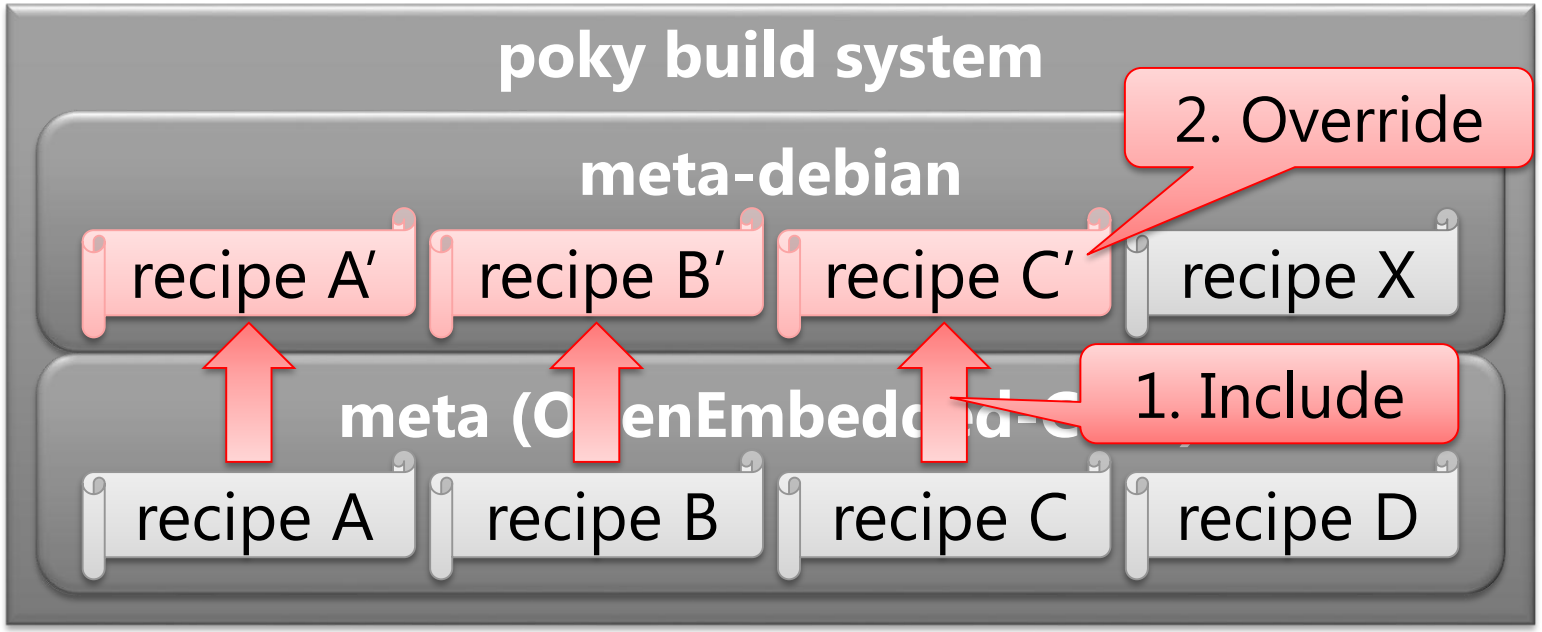
How should we create recipes in a layer?

- **From scratch?**
  - Often takes time!
  - Why?
    - Need to create patches for supporting cross-build in poky
- **We should follow the existing OE-Core recipes**
  - How?



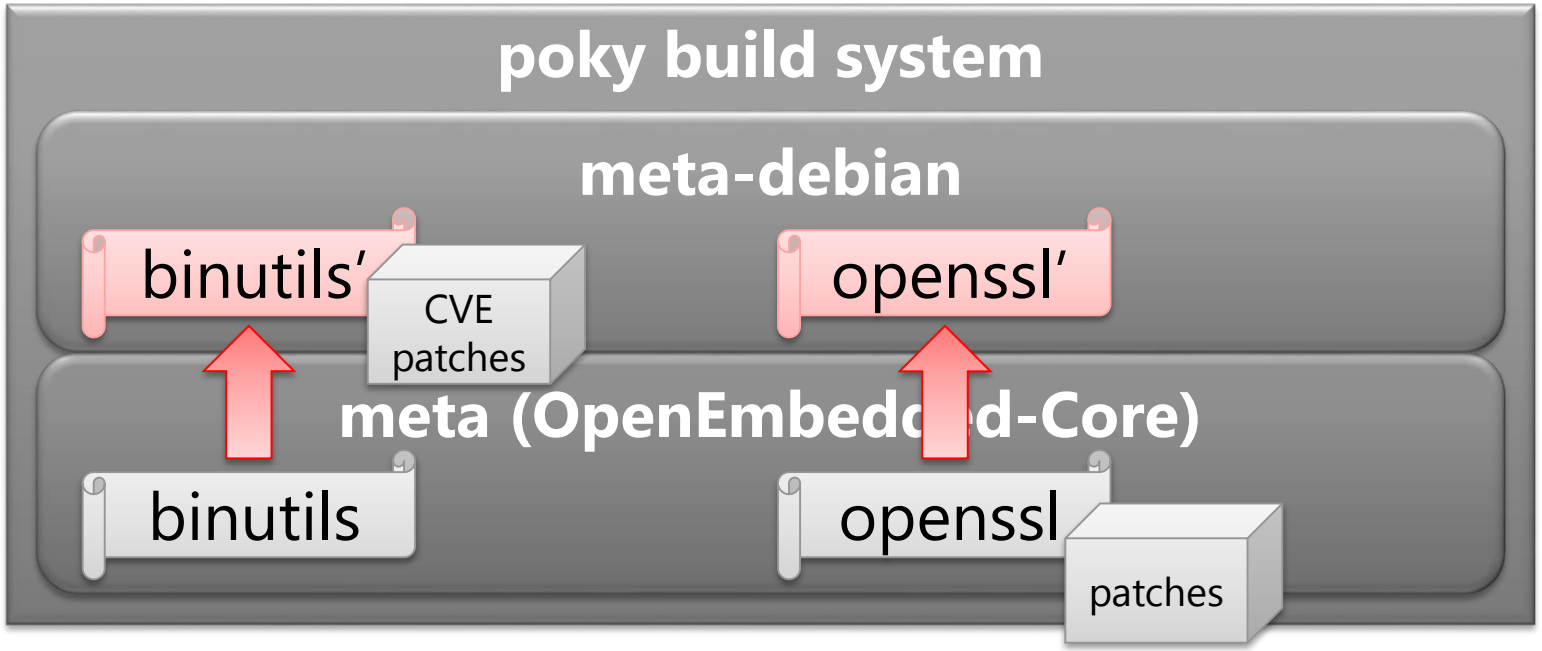
Method 1: "Include" OE-Core recipes

- **We used to use this method before[2]**
- **Unsuitable for our case ☹**
  - Difficult to override some variables and functions
    - Ex: already appended (_append) or prepended (_prepend) data
  - Automatically follow "unneeded" OE-Core updates against our will
    - Ex: Shown in the next slides



Method 1: "Include" OE-Core recipes

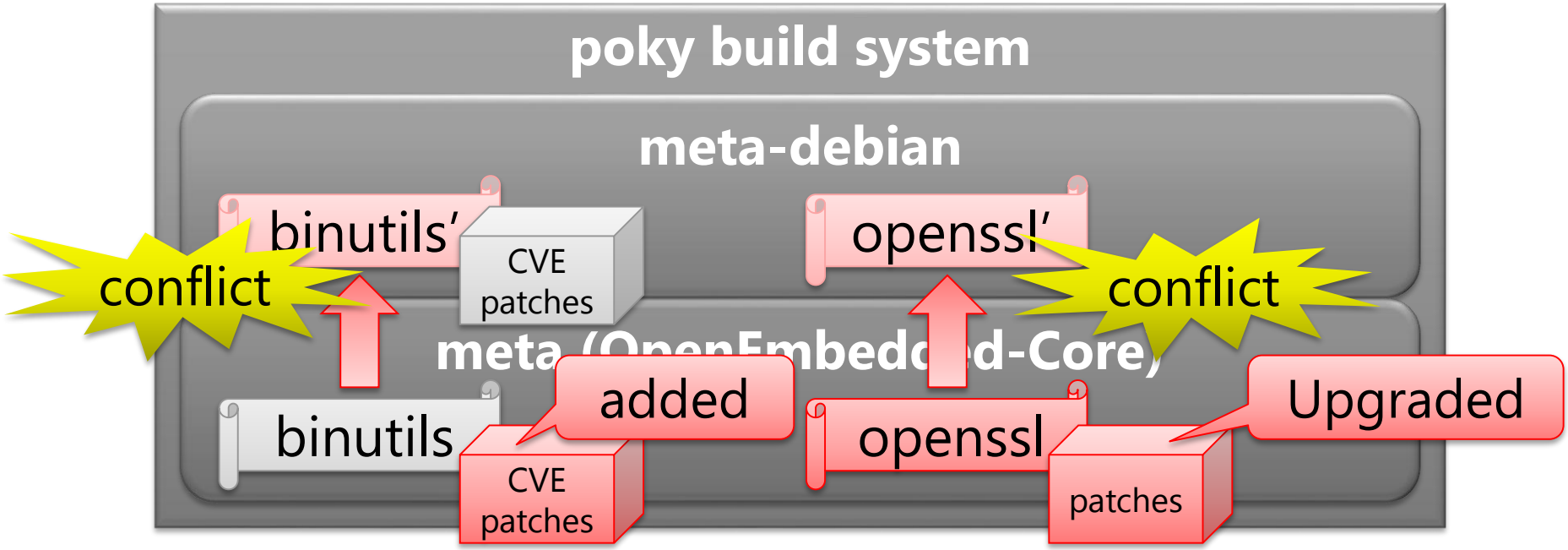
- **binutils**
- **openssl**



Method 1: "Include" OE-Core recipes

- **binutils**
  - Security patches applied twice
- **openssl**
  - Target version was upgraded, and patches also upgraded
  - Some upgraded patches conflict with Debian source

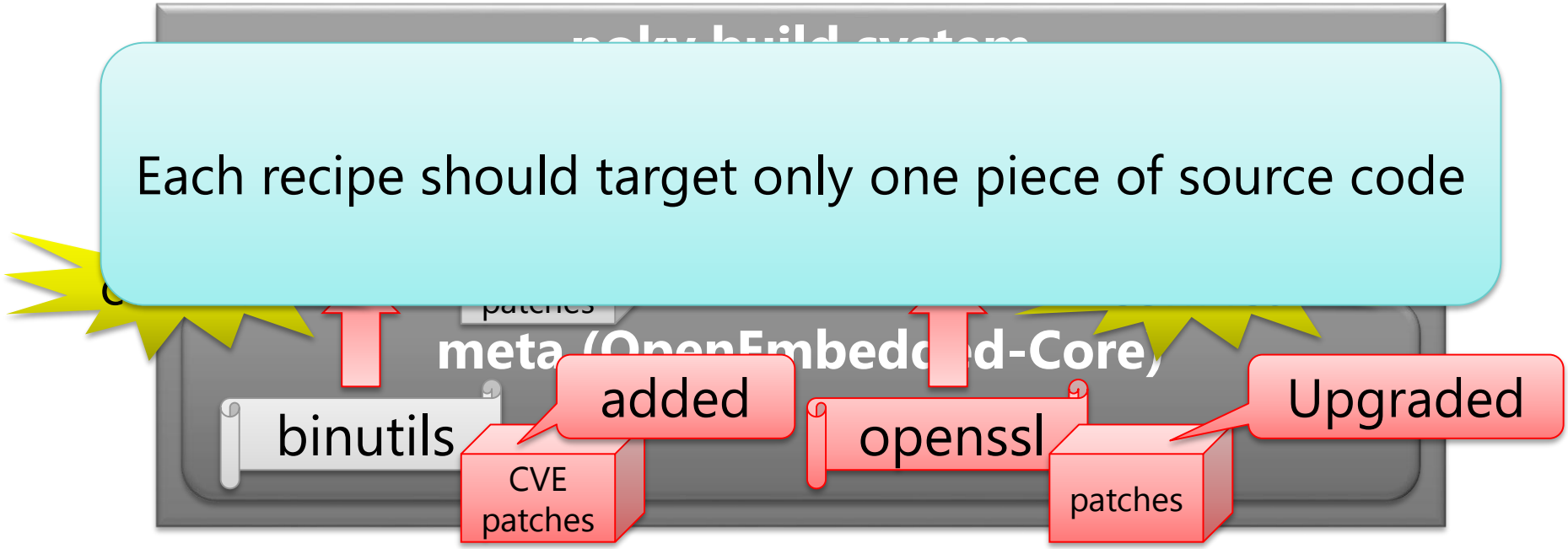
Difficult to maintain ☹



Method 1: "Include" OE-Core recipes

- **binutils**
  - Security patches applied twice
- **openssl**
  - Target version was upgraded
  - Some upgraded patches conflict with Debian source

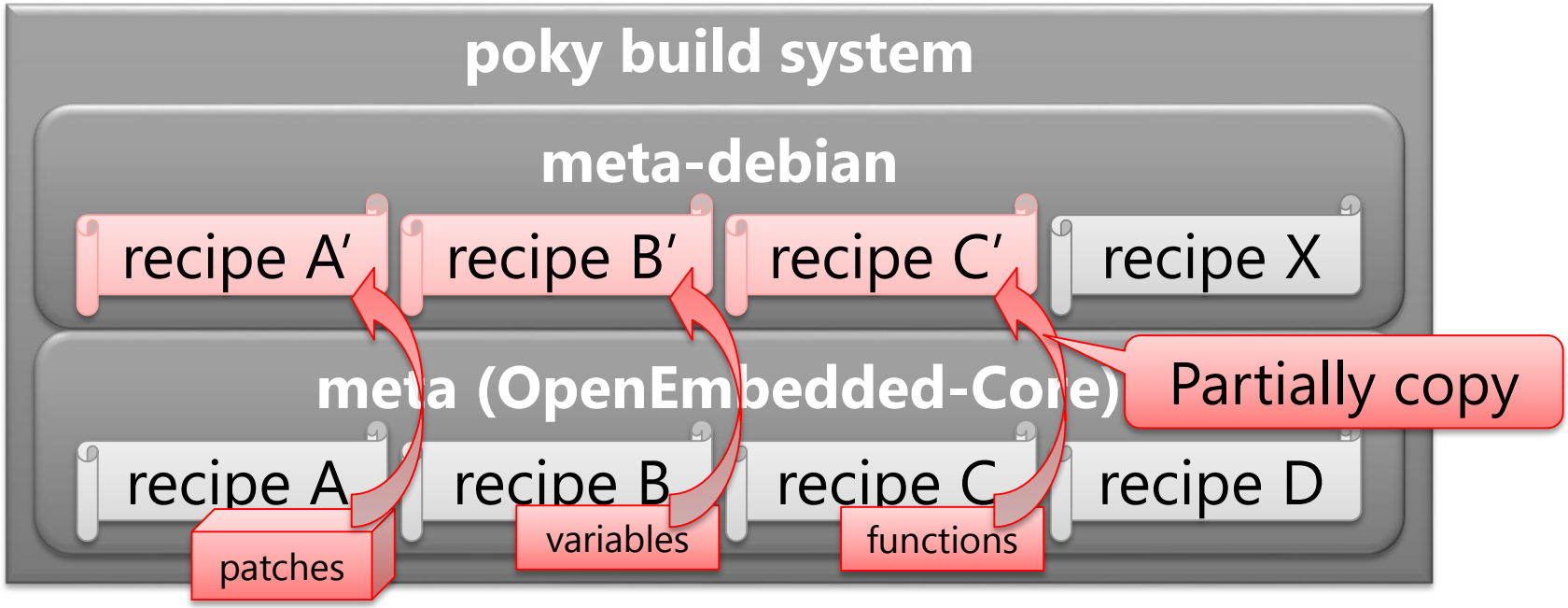
Difficult to maintain ☹



Method 2: Copy OE-Core recipes

our solution

- **Create recipes from scratch using Debian source packages**
- **Copy (re-use) only essential data from OE-Core**
  - patches, variables, functions for supporting cross-build



How should we implement recipes?

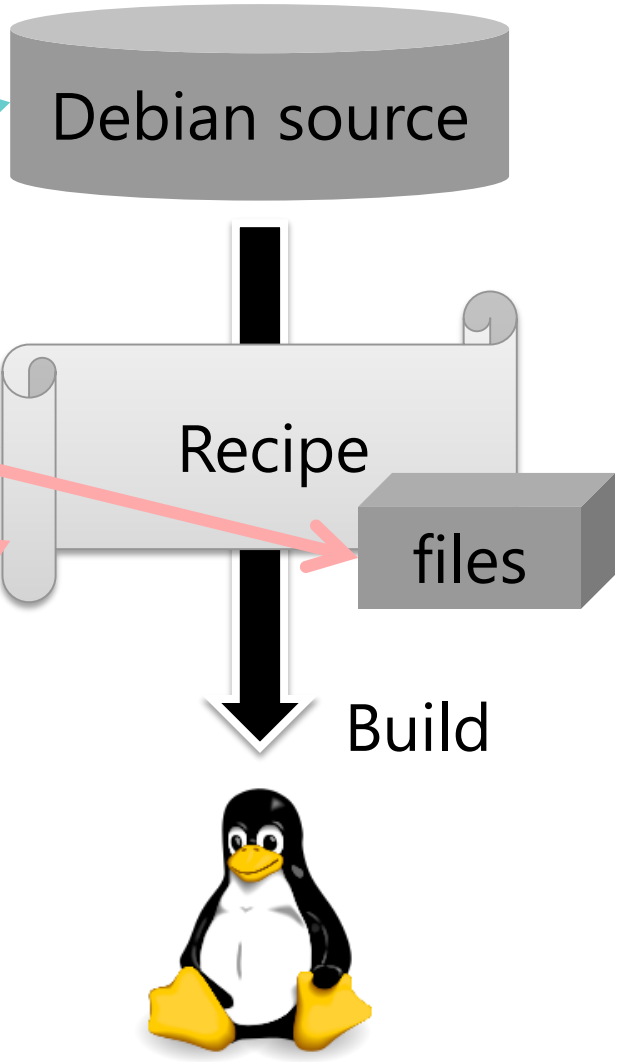
- LICENSE information
- Required files
  - Source code
  - initscripts, configs
  - Patches
- Configure commands & options
- Compile commands & options
- Installed files and paths
- How to package files
- Dependencies between others

Method 1: re-use OE-Core (poky-debian[2])

- LICENSE information
- Required files
  - Source code
  - initscripts, configs
  - Patches
- Configure commands & options
- Compile commands & options
- Installed files and paths
- How to package files
- Dependencies between others

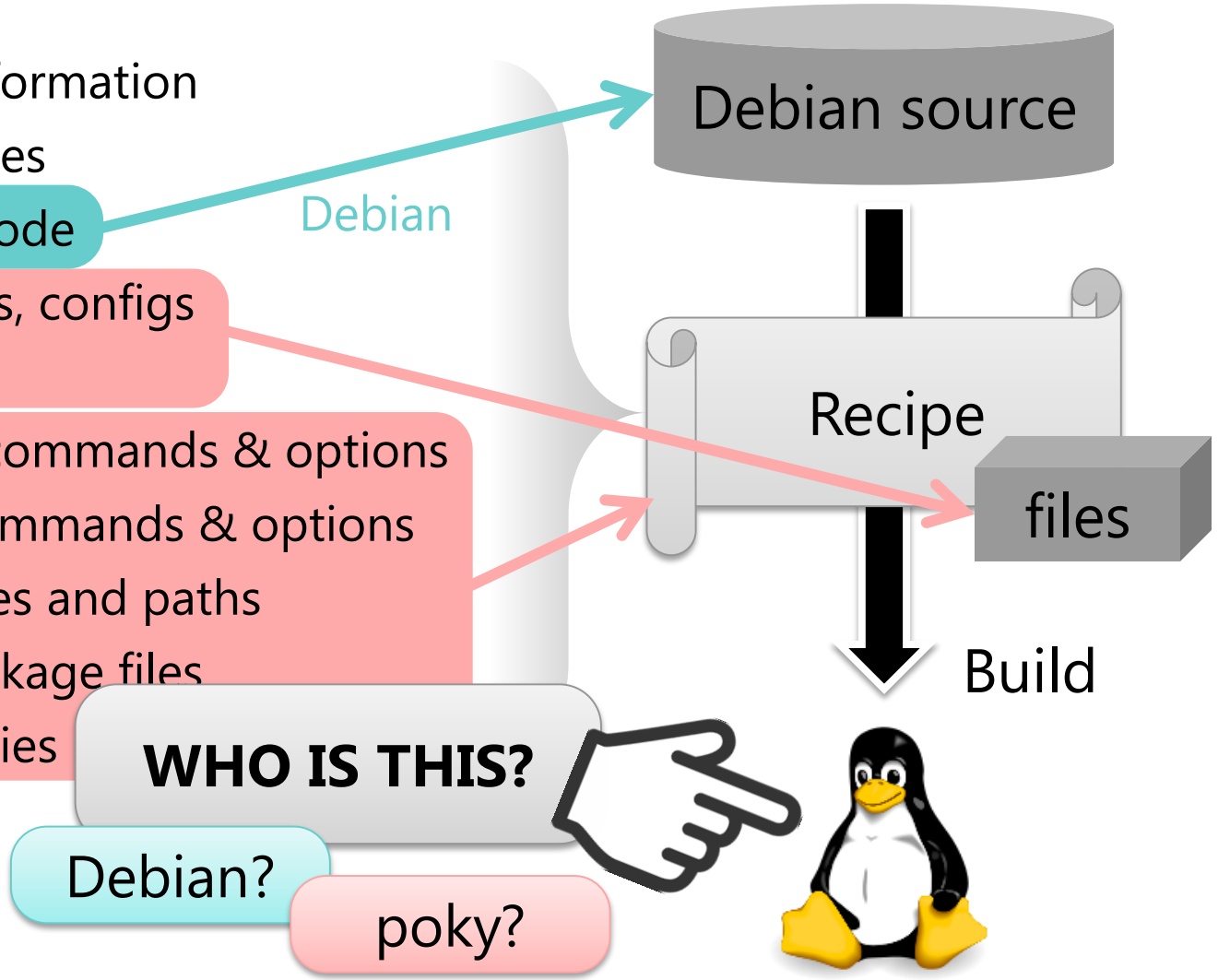
Debian

OE-Core based



Method 1: re-use OE-Core (poky-debian[2])

- LICENSE information
- Required files
  - Source code
  - initscripts, configs
  - Patches
- Configure commands & options
- Compile commands & options
- Installed files and paths
- How to package files
- Dependencies



Method 1: re-use OE-Core (poky-debian[2])

- **Bad results: conflicts of two distributions**
  - Compile fails
    - Cause: missing configure options that Debian source requires
  - Some programs fail to call commands or load data file
    - Cause: installation paths differ from Debian's
- **Cannot be used like Debian**

We should define our "policy" for creating recipes

Policies for creating recipes

our solution

- **By default, follow Debian's packaging**
  - i.e. debian/rules
  - For getting good affinity with Debian sources
- **Customize for embedded system if necessary**
  - Disable features
  - Remove dependencies
- **Re-use only essential data from OE-Core for supporting cross-compile**
  - See "Method 2: Copy OE-Core recipes"

Method 2: Follow Debian's packaging

our solution

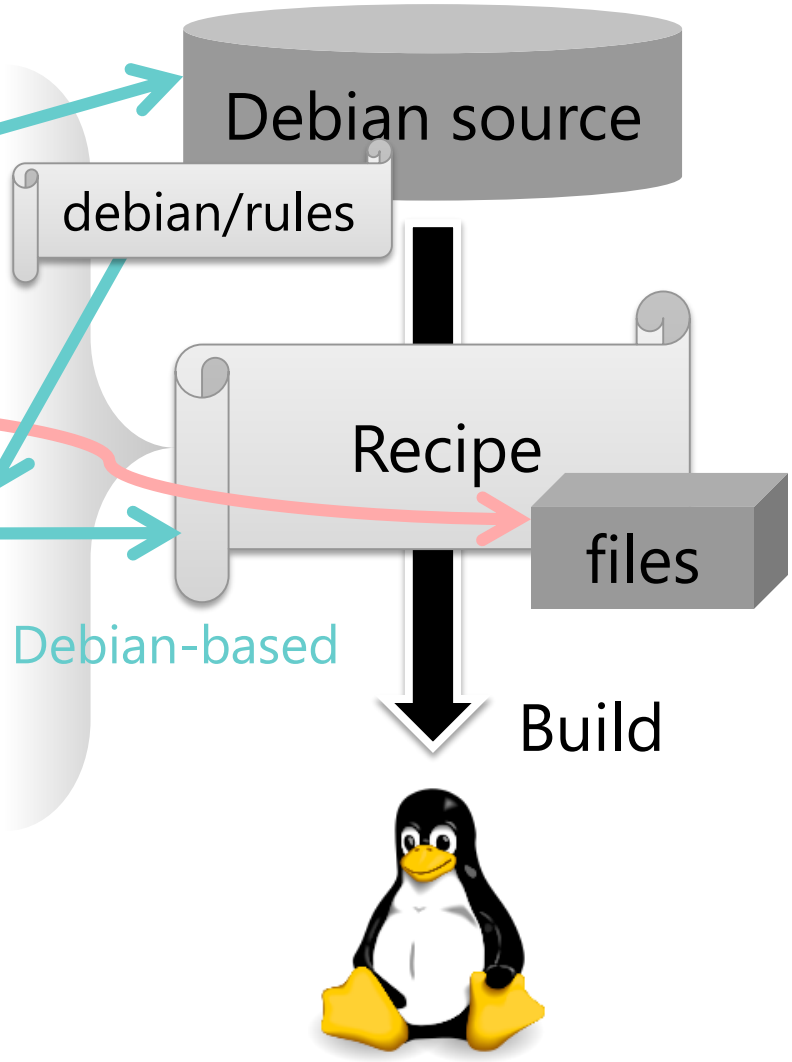
- LICENSE information
- Required files
  - Source code
  - initscripts, configs
  - Patches
- Configure commands & options
- Compile commands & options
- Installed files and paths
- How to package files
- Dependencies between others

Customize for embedded

Debian

OE-Core based

Debian-based



How to create recipes (Sample: zlib)

meta-debian/recipe-debian/zlib/zlib_debian.bb

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe_runmake
}
do_install() {
        oe_runmake DESTDIR=${D} install
}
do_install_append_class-target() {
        mkdir -p ${D}/${base_libdir}
        mv ${D}/${libdir}/libz.so.* ${D}/${base_libdir}
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

Step1: Add recipe revision

```
PR = "r0"
inherit debian pa

LICENSE = "Zlib"
LIC_FILES_CHKSUM
"file://zlib.h;b                                    8b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe_runmake
}
do_install() {
        oe_runmake DESTDIR=${D} install
}
do_install_append_class-target() {
        mkdir -p ${D}/${base_libdir}
        mv ${D}/${libdir}/libz.so.* ${D}/${base_libdir}
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

- Define recipe revision: ${PR}
- Increment every update

Step2: Inherit debian-package.bbclass

```
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM =
"file://zlib.h;beginl

SRC_URI += "file://remo

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe_runmake
}
do_install() {
        oe_runmake DESTDIR=${D} install
}
do_install_append_class-target() {
        mkdir -p ${D}/${base_libdir}
        mv ${D}/${libdir}/libz.so.* ${D}/${base_libdir}
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

Debian based

- Setup Debian source package
  - Define SRC_URI
  - Apply Debian's patches (do_debian_patch)

Step3: Add license information

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "fil://re

do_configure() {
        ./configu
}
do_compile () {
        oe_runmake
}
do_install() {
        oe_runmake
}
do_install_append_
        mkdir -p ${
        mv ${D}/${lib
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

- LICENSE: License name
  - Common license names are found in meta/files/common-licenses
- LIC_FILES_CHKSUM: Checksum of the license text
  - Usually found in COPYING, LICENSE, or header of source files (.c, .h)

Step4: Append patches

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_co figure() {

}
do_compile() {
        oe_
}
do_install() {
        oe_runmake DESTDIR=${D} install
}
do_install_append_class-target() {
        mkdir -p ${D}/${base_libdir}
        mv ${D}/${libdir}/libz.so.* ${D}/${base_libdir}
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

OE-Core based

- Add patches into SRC_URI
  - Necessary for being built in cross-compile environment
  - Copied from OE-Core (or from scratch)

Step5: Define configure options

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe
}
do_instal
        oe
}
do_install
        mk
        mv ${D}/${libdir}/libz.so.* ${D}/${base_libdir}
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

- Define configure commands
  - The same options as debian/rules
  - Some features should be disabled for embedded

Debian based

Step6: Define compile and install commands

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe_runmake
}
do_install() {
        oe_runmake DESTDIR=${D} install
}
do_install_append_class-target() {
        mkdir -p ${D}/
        mv ${D}/${li
        tmp=`readlink
        ln -sf ../../$
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

- Define compile & install commands
  - autotools.bbclass often replaces them

Additional Steps: Change library paths

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe_r
}
do_install()
        _i
}

do_install_append_class-target() {
        mkdir -p ${D}/${base_libdir}
        mv ${D}/${libdir}/libz.so.* ${D}/${base_libdir}
        tmp=`readlink ${D}/${libdir}/libz.so`
        ln -sf ../../${base_libdir}/$tmp ${D}/${libdir}/libz.so
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

Debian based

Move run-time libraries to the same directory as Debian

Additional Steps: Change package name

```
PR = "r0"
inherit debian-package

LICENSE = "Zlib"
LIC_FILES_CHKSUM = \
"file://zlib.h;beginline=4;endline=23;md5=fde612df1e5933c428b73844a0c494fd"

SRC_URI += "file://remove.ldconfig.call.patch"

do_configure() {
        ./configure --shared --prefix=${prefix} --libdir=${libdir}
}
do_compile () {
        oe_runmake
}
do_install() {
        oe_runmake DESTDIR=${D} install
}
do_install_app
        mkdir
        mv ${D
        tmp=`
        ln
}

DEBIANNAME_${PN}-dbg       = "${PN}1g-dbg"
DEBIANNAME_${PN}-staticdev = "${PN}1g-staticdev"
DEBIANNAME_${PN}-dev       = "${PN}1g-dev"
DEBIANNAME_${PN}-doc       = "${PN}1g-doc"
DEBIANNAME_${PN}           = "${PN}1g"
```

Debian based

- Change the default binary package name to Debian's
- "libz" => "zlib1g"

Build results (zlib packages)

Debian 8.0 jessie

zlib1g
- /lib/i386-linux-gnu/libz.so.1
- /lib/i386-linux-gnu/libz.so.1.2.8
- /usr/share/doc/zlib1g/...

zlib1g-dev
- /usr/include/i386-linux-gnu/zconf.h
- /usr/include/zlib.h
- /usr/lib/i386-linux-gnu/libz.so
- /usr/lib/i386-linux-gnu/pkgconfig/zlib.pc
- /usr/share/doc/...
- /usr/share/man/...
- /usr/lib/i386-linux-gnu/libz.a

zlib1g-dbg

zlib1g-udeb

lib32z1* | lib64z1* | libn32z1*

meta-debian

zlib1g
- /lib/libz.so.1
- /lib/libz.so.1.2.8

zlib1g-dev
- /usr/include/zconf.h
- /usr/include/zlib.h
- /usr/lib/libz.so
- /usr/lib/pkgconfig/zlib.pc

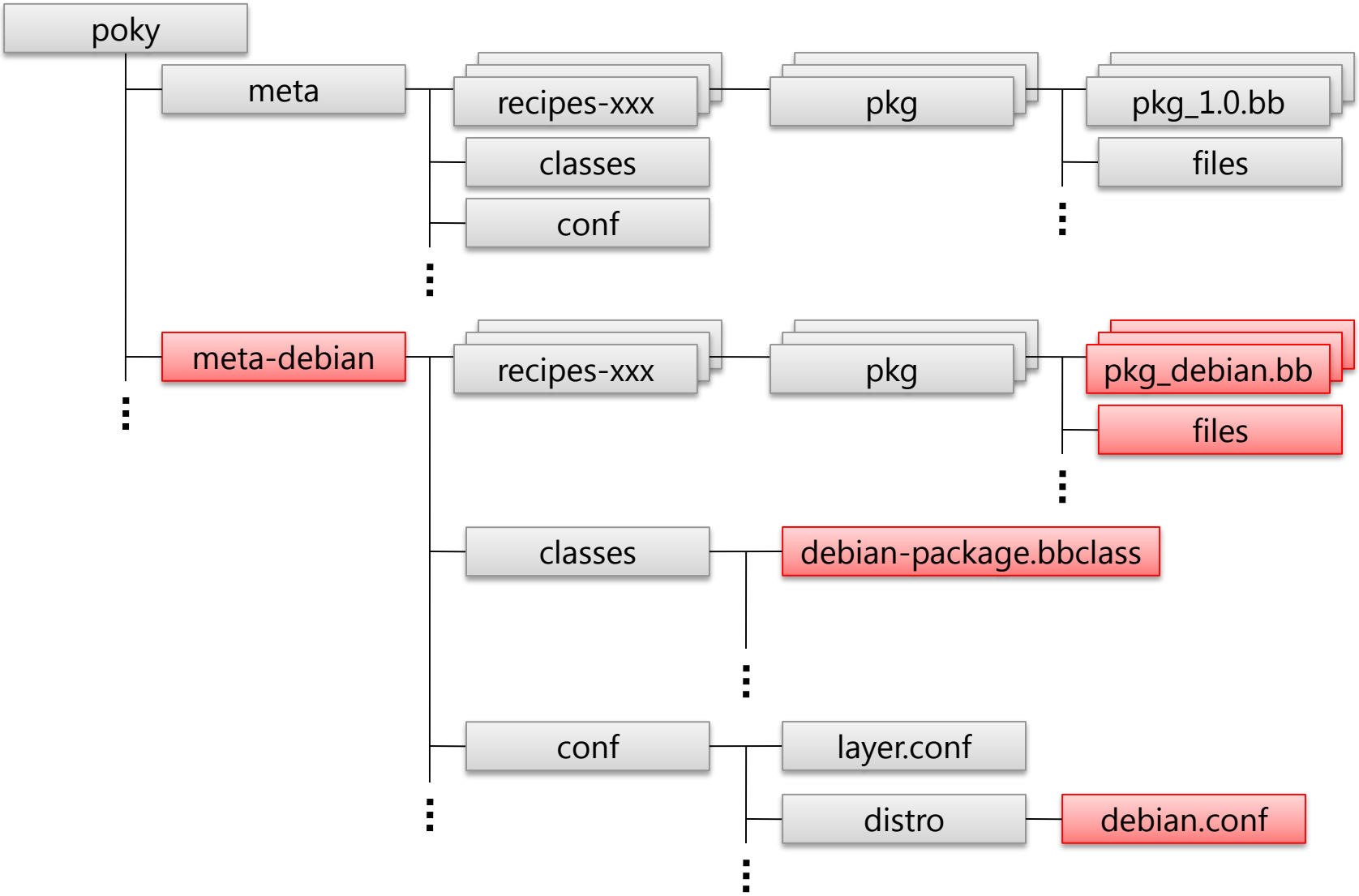
zlib1g-doc
- /usr/share/man/...

zlib1g-staticdev
- /usr/lib/libz.a

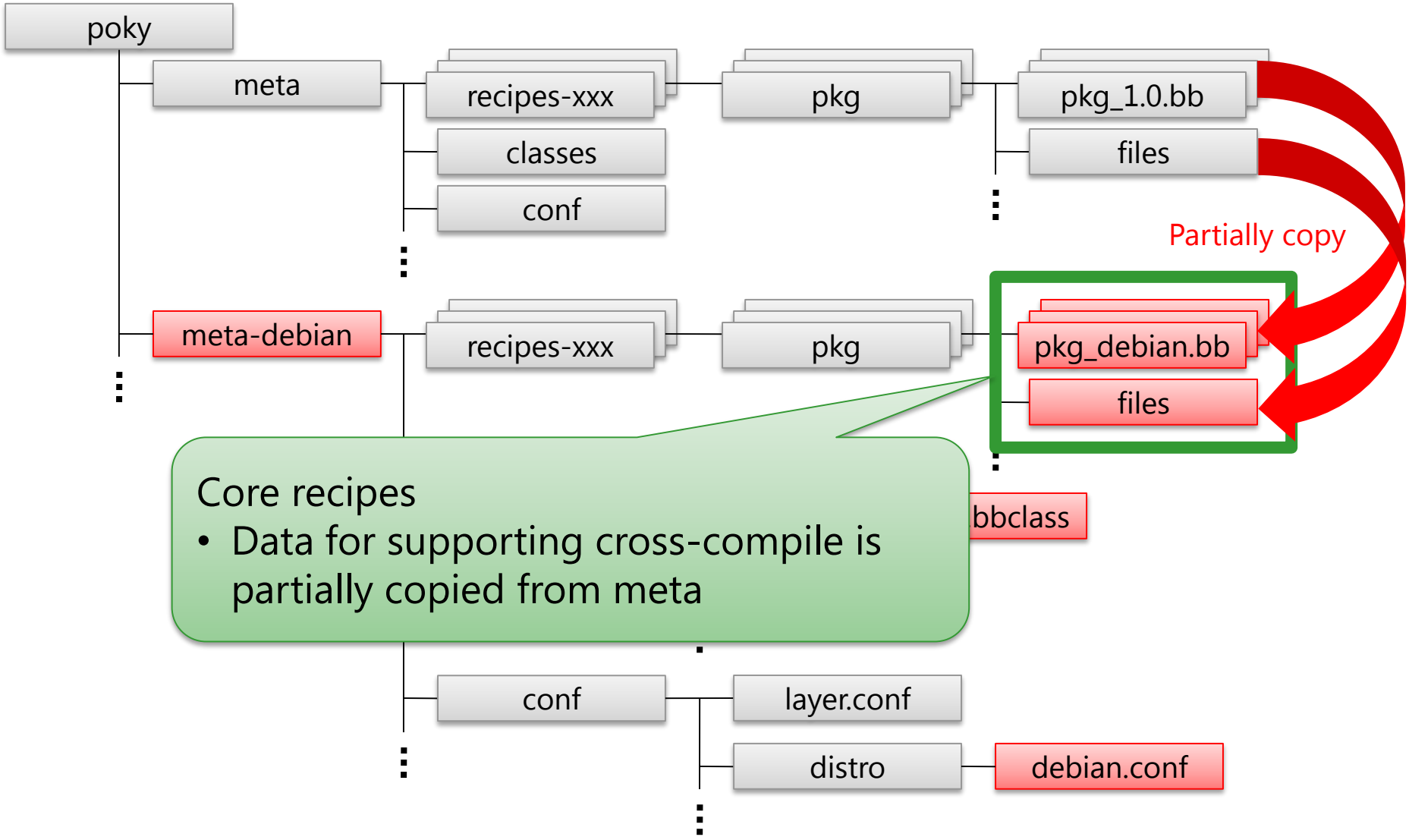
zlib1g-dbg

Ignore non-essential files

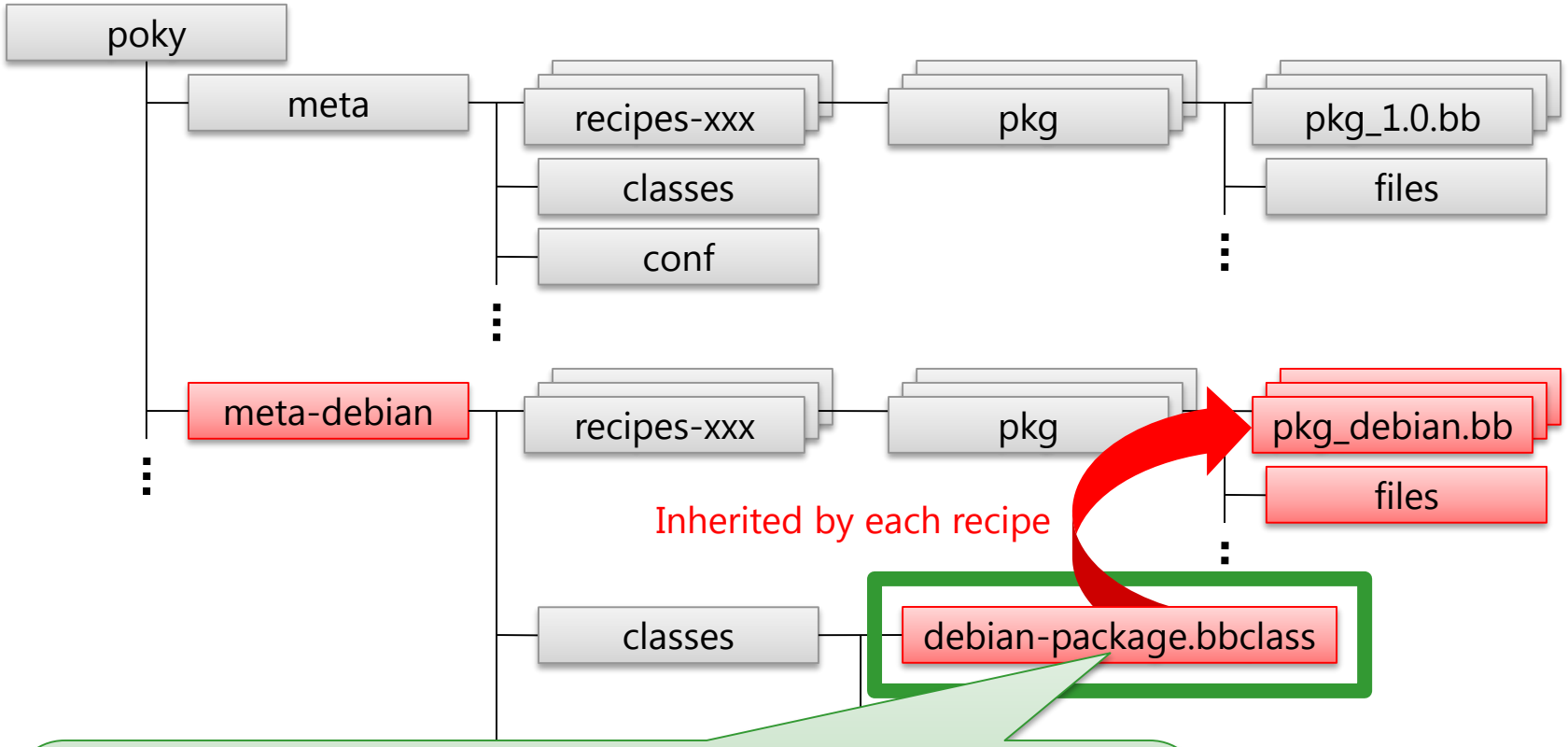
Directory structure



Directory structure



Directory structure

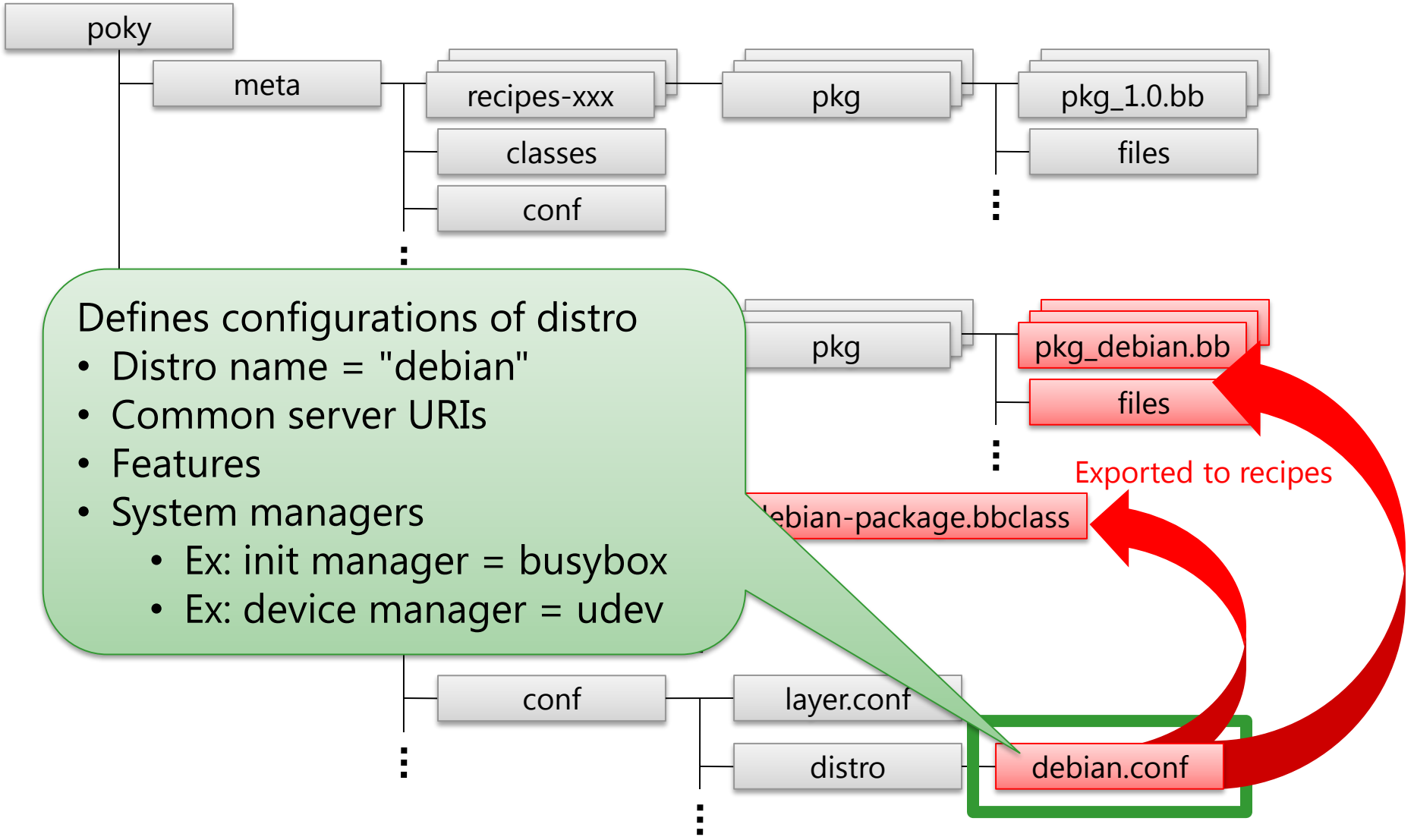


Provides debian specific functions and variables

- Fetch a source package automatically
- Apply Debian's patches automatically
  - debian/patches/*

debian.conf

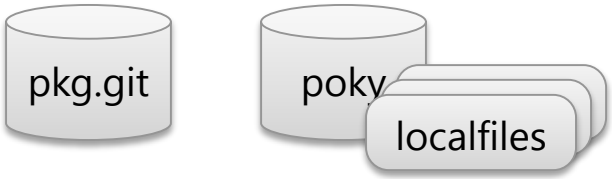
Directory structure



Build flow

bitbake tasks

do_fetch()
do_unpack()
do_debian_patch()
do_patch()
do_configure()
do_compile()
do_install()
do_package()
.....



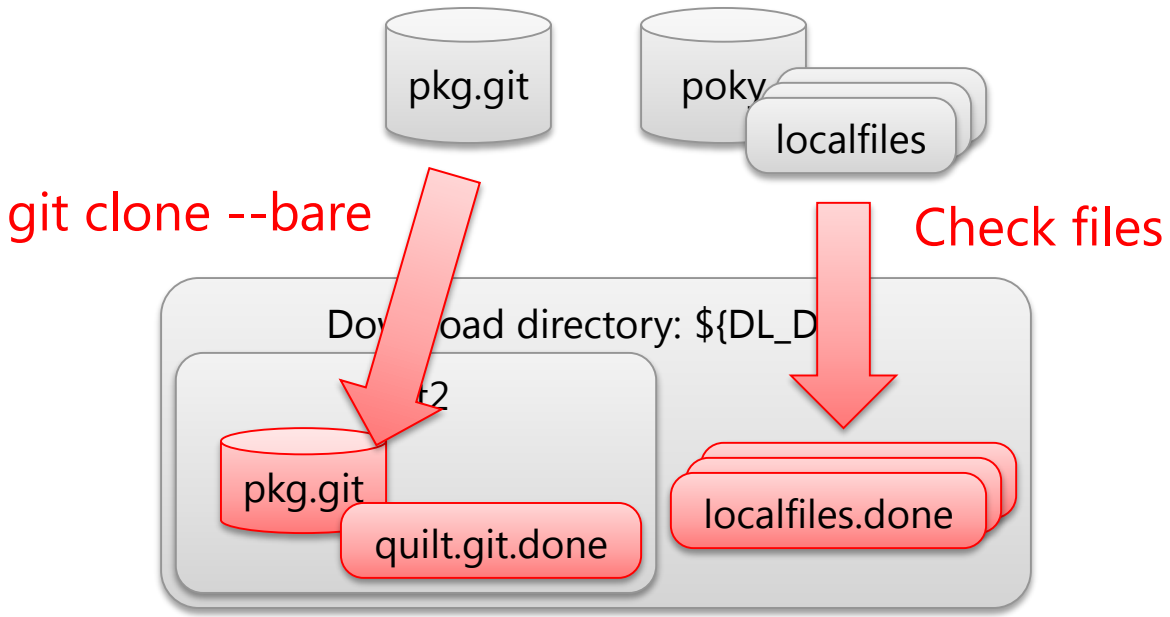
Download directory: ${DL_DIR}

Working directory: ${WORKDIR}

Build flow

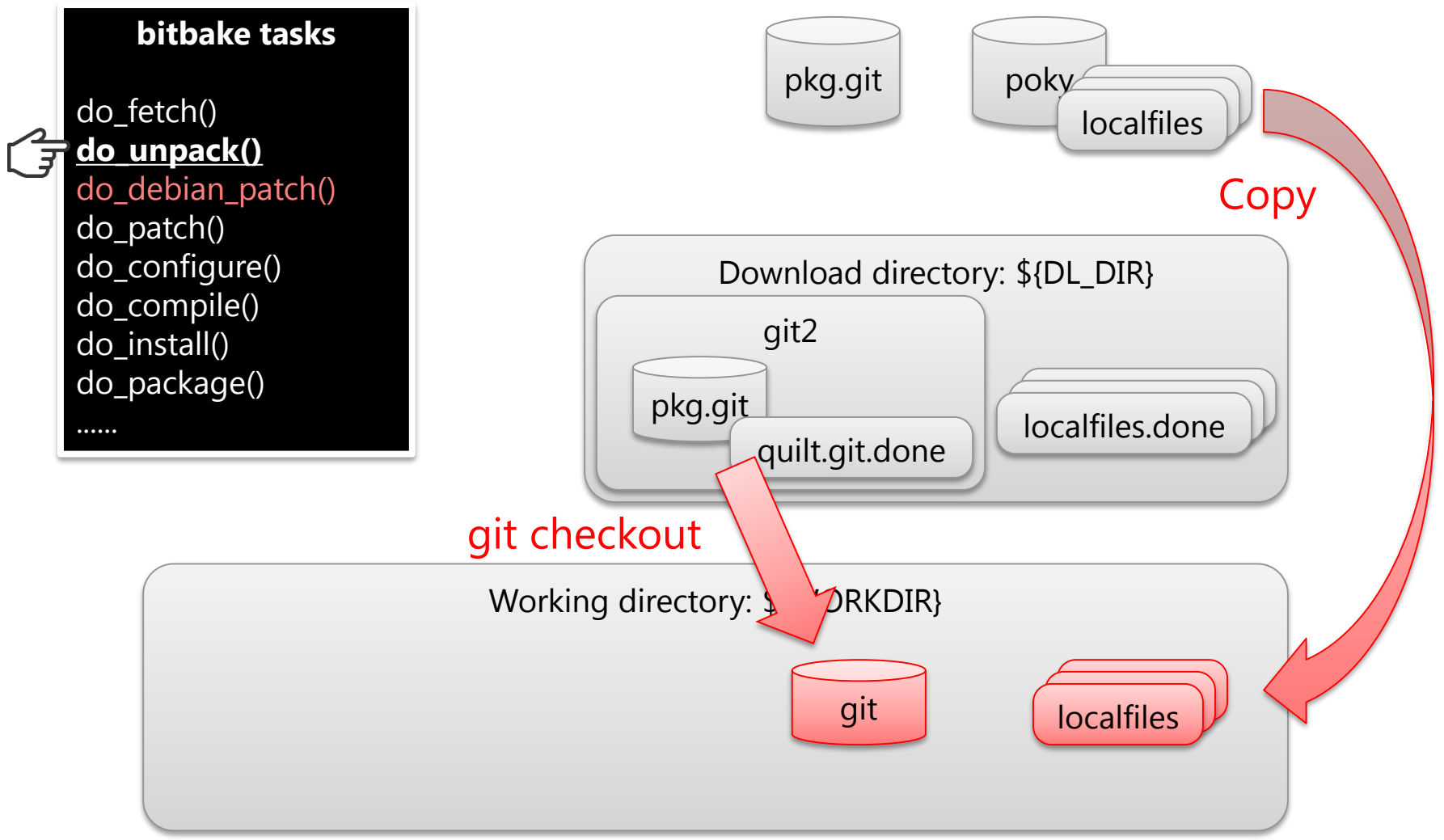
bitbake tasks

do_fetch()
do_unpack()
do_debian_patch()
do_patch()
do_configure()
do_compile()
do_install()
do_package()
.....

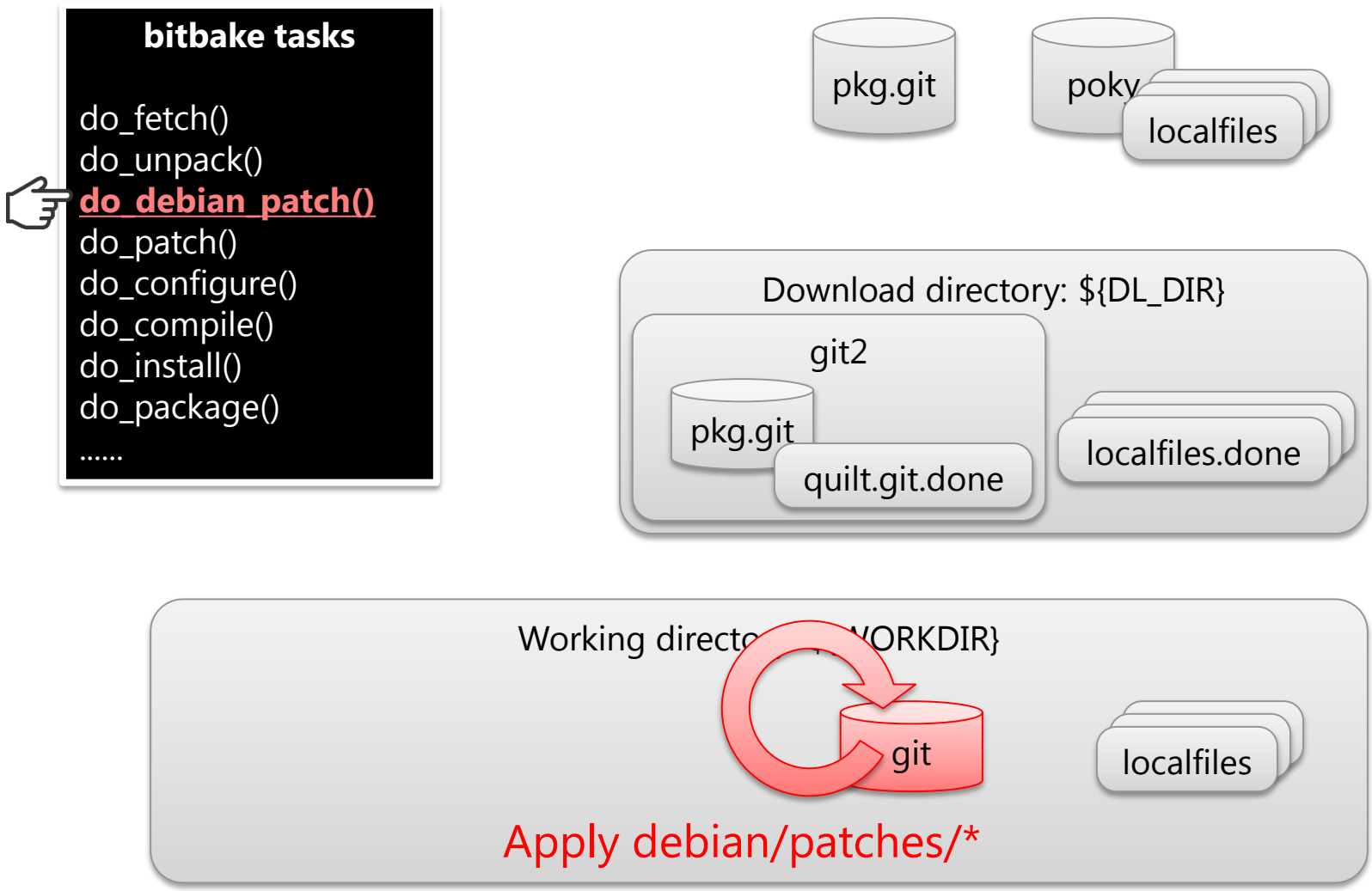


Working directory: ${WORKDIR}

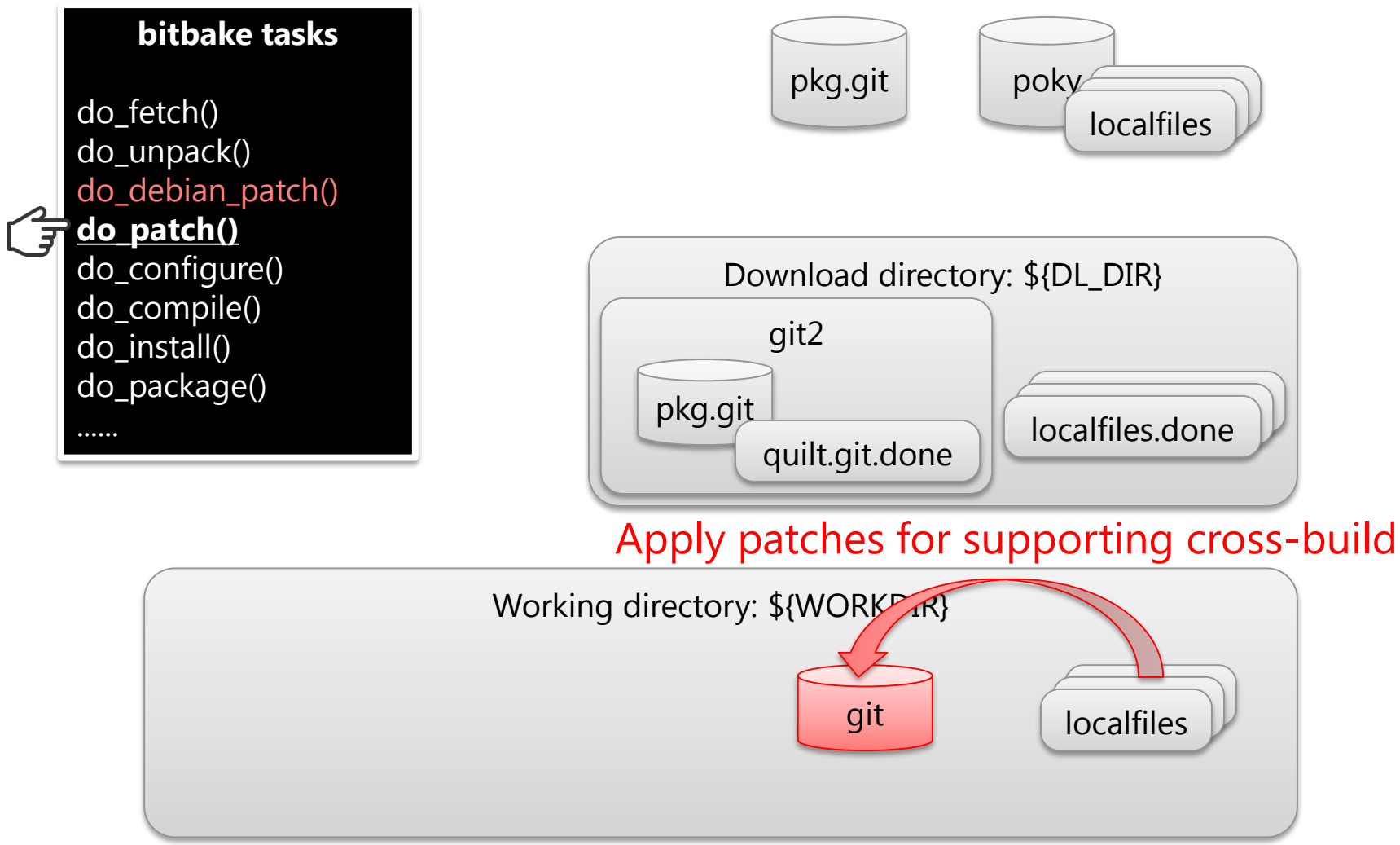
Build flow



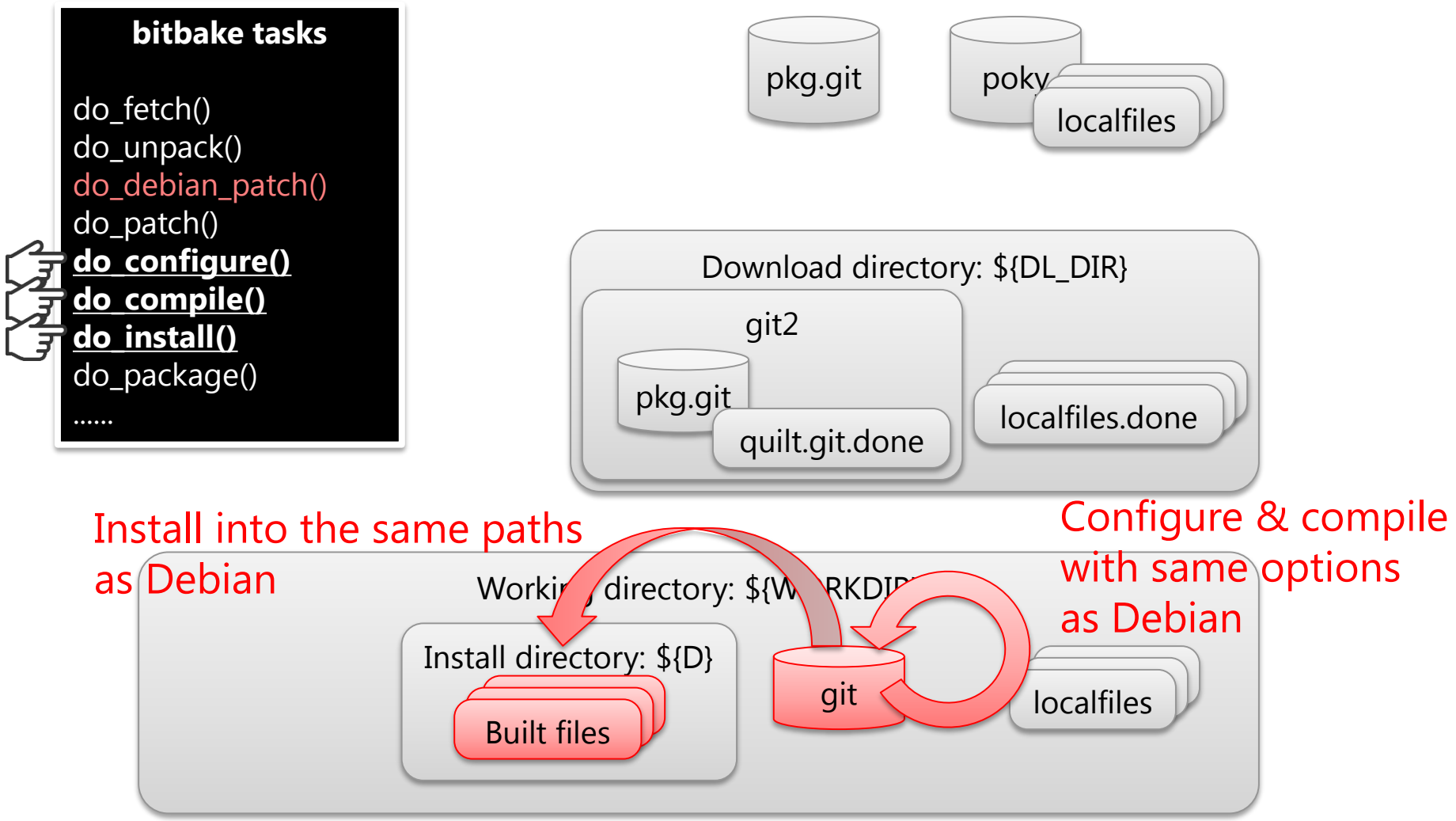
Build flow



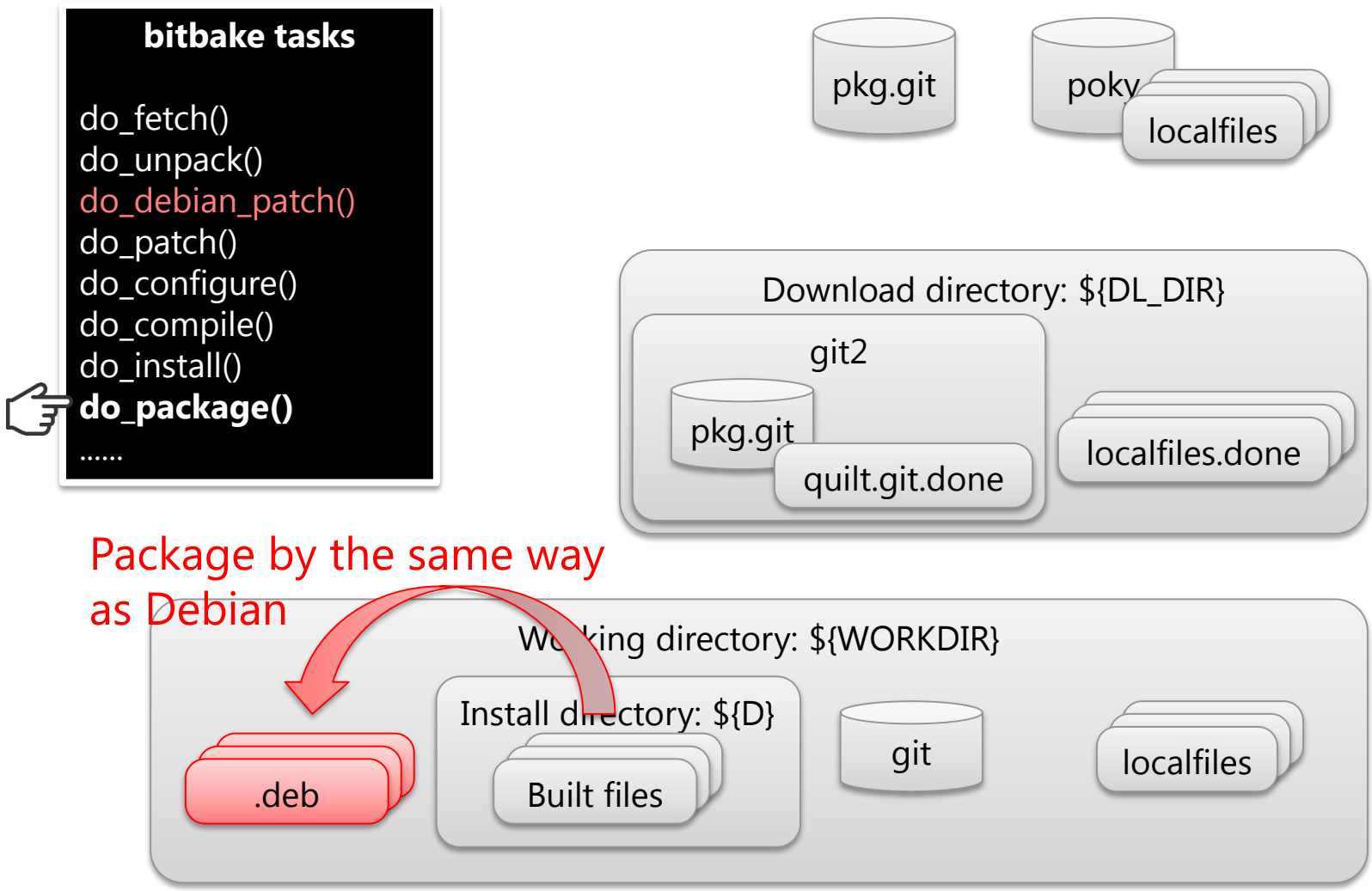
Build flow



Build flow



Build flow



Conclusions

- **What is meta-debian ?**
  - Metadata for building embedded Linux systems using Debian source packages
  - Implemented as an independent layer of OpenEmbedded-Core
- **meta-debian is intended to provide**
  - Wide embedded CPU support
  - Stability
  - Long-term support
  - Fully customizable Linux

Conclusions

- **Policies for creating recipes**
  - Debian based configs & packaging + customization for embedded
    - For getting affinity with Debian sources
  - Re-use OE-Core data for supporting cross-build
- **Examples**
  - How to build & run a tiny Linux image
  - How to create recipes

Current state

- **Supported CPUs**
  - x86 32bit
  - x86 64bit
  - ARM
  - PowerPC
- **Kernel**
  - LTSI
- **User land**
  - busybox-based tiny system
  - Number of available packages: about 80
    - We are now implementing more recipes to support other packages

Future work

- **Support more features and packages (over 200)**
  - LTSI + RT kernel, X server, Qt5, etc.
- **Support SDK (meta-toolchain)**
- **Support more embedded boards**
- **Testing: improve the quality of the system and packages**
  - LTP, LTP-DDT, POSIX test suite, ptest, etc.
  - Contribute to the LTSI test project (JTA) [3]
- **Keep following updates of poky and Debian**

Please give us some feedback

- **E-mail**
  - yoshitake.kobayashi@toshiba.co.jp
  - kazuhiro3.hayashi@toshiba.co.jp
- **Mailing list**
  - https://groups.google.com/forum/#!forum/meta-debian
- **Repository**
  - https://github.com/meta-debian/meta-debian.git

Questions

References

1. **EG-5000 Automatic Ticket Gate with High Reliability and Scalability**
   - http://www.toshiba.co.jp/tech/review/2010/10/65_10pdf/f05.pdf
2. **Poky meets Debian: Understanding How to Make an Embedded Linux by Using an Existing Distribution's Source Code**
   - http://events.linuxfoundation.org/sites/events/files/slides/ELC2015-YOSHI-PokyDebian_0.pdf
3. **LTSI Test Project**
   - http://ltsi.linuxfoundation.org/ltsi-test-project
4. **Yocto Project Manuals**
   - http://www.yoctoproject.org/docs/1.6/poky-ref-manual/poky-ref-manual.html
   - http://www.yoctoproject.org/docs/1.6/dev-manual/dev-manual.html
   - http://www.yoctoproject.org/docs/1.6/bitbake-user-manual/bitbake-user-manual.html


TOSHIBA
Leading Innovation

remove.ldconfig.call.patch

```
When /etc/ld.so.cache is writeable by user running bitbake then it creates
invalid cache
(in my case libstdc++.so cannot be found after building zlib(-native) and I
have to call
touch */libstdc++.so && /sbin/ldconfig to fix it.

So remove ldconfig call from make install-libs

Upstream-Status: Inappropriate [disable feature]

diff -uNr zlib-1.2.6.orig/Makefile.in zlib-1.2.6/Makefile.in
--- zlib-1.2.6.orig/Makefile.in 2012-01-28 23:48:50.000000000 +0100
+++ zlib-1.2.6/Makefile.in      2012-02-13 15:38:20.577700723 +0100
@@ -199,7 +199,6 @@
          rm -f $(DESTDIR)$(sharedlibdir)/$(SHAREDLIB)
$(DESTDIR)$(sharedlibdir)/$(SHAREDLIBM); \
          ln -s $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)/$(SHAREDLIB); \
          ln -s $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBM); \
-         ($(LDCONFIG) || true)  >/dev/null 2>&1; \
        fi
        cp zlib.3 $(DESTDIR)$(man3dir)
        chmod 644 $(DESTDIR)$(man3dir)/zlib.3
```

recipes-extended/newt/files/cross_ar.patch

```
...
Makefile.in  |    3 ++-
 configure.ac |    4 ++++
 2 files changed, 6 insertions(+), 1 deletion(-)

--- a/Makefile.in
+++ b/Makefile.in
@@ -7,6 +7,7 @@ CFLAGS = @CFLAGS@
 LDFLAGS = @LDFLAGS@
 CPPFLAGS = -D_GNU_SOURCE @CPPFLAGS@
 GNU_LD = @GNU_LD@
+AR = @AR@

 VERSION = @VERSION@
 TAG = r$(subst .,-,$(VERSION))
@@ -95,7 +96,7 @@ whiptcl.so: $(WHIPTCLOBJS) $(LIBNEWTSH)
        $(CC) -shared $(SHCFLAGS) $(LDFLAGS) -o whiptcl.so $(WHIPTCLOBJS) -
L. -lnewt  $(LIBTCL) -lpopt $(LIBS)

 $(LIBNEWT): $(LIBOBJS)
-       ar rv $@ $^
+       $(AR) rv $@ $^

 newt.o $(SHAREDDIR)/newt.o: newt.c Makefile
...
```

qt4-embedded_git.bb

```
require qt4-libs.inc

PR = "r0"

QT_CONFIG_FLAGS = " \
-v \
-embedded ${QT_ARCH} \
-release \
-opensource \
-make libs \
-nomake tools \
-nomake examples \
-nomake demos \
...
```